```
%% Machine Learning
% Lab 1: Linear Regression with One variable
% ---- Profit in City -----
%{
In this part of this exercise, you will implement linear regression with
one variable to predict profits for a food truck. Suppose you are the CEO
of a restaurant franchise and are considering diferent cities for opening
a new outlet. The chain already has trucks in various cities and you have
data for profits and populations from the cities. You would like to use
this data to help you select which city to expand to next.
%}

%% Initialization ============================================
clear ; close all; clc

fprintf('Plotting Data...\n')
data=load('ex1data1.txt');
fprintf('First 5 element of data:\n')
data(1:5, :)

X = data(:, 1);
Y = data(:, 2);
m = length(Y);  % number of training examples

plot(X, Y, 'rx', 'MarkerSize', 10);       % Plot the data
ylabel('Profit in $10,000s');             % Set the y-axis label
xlabel('Population of City in 10,000s');  % Set the x-axis label


%% Add Bias ==================================================
X = [ones(m, 1), data(:,1)]; % Add a column of ones to X
w = zeros(2,1);              % Initialize fitting parameter

% Some gradient descent settings
epochs = 1500;
learning_rate = 0.01;


%% Testing the Cost Function / Error Function ... ==========
fprintf('\nTesting the cost function (Error function) ...\n')

% compute and display initial cost
C = computeCost(X,Y,w);

fprintf('\tWith theta = [0 ; 0]');
fprintf('\n\tCost computed = %f\n', C);
fprintf('\tExpected cost value (approx) 32.07\n');

% further testing of the cost function
C = computeCost(X, Y, [-1 ; 2]);  %modified weights
fprintf('\n\tWith theta = [-1 ; 2]');
fprintf('\n\tCost computed = %f', C);
fprintf('\n\tExpected cost value (approx) 54.24\n');
```

```matlab
%% Gradient Descent ... ====================================
w = gradientDescent(X, Y, w, learning_rate, epochs);

% print weights to screen
fprintf('\nWeights found by gradient descent:\n');
fprintf('%f\n', w);
fprintf('Expected weights values (approx)\n');
fprintf(' -3.6303\n1.1664\n\n');


%% Plot the linear fit ====================================
hold on; % keep previous plot visible
plot(X(:,2), X*w, '-') % X*w is Y_estimated
legend('Training data', 'Linear regression')
hold off % don't overlay any more plots on this figure


%%  Predict values for population sizes of 35,000 and 70,000
predict1 = [1, 3.5] *w;
fprintf('For population = 35,000, we predict a profit of %f\n',...
    predict1*10000);
predict2 = [1, 7] * w;
fprintf('For population = 70,000, we predict a profit of %f\n',...
    predict2*10000);


%% Vizualising Cost function ================================
fprintf('Visualizing C(w_0, w_1) ...\n')

% Grid over which we will calculate Cost Function (C)
w0_vals = linspace(-10, 10, 100);
w1_vals = linspace(-1, 4, 100);

% initialize C_vals to a matrix of 0's
C_vals = zeros(length(w0_vals), length(w1_vals));

% Fill out C_vals
for i = 1:length(w0_vals)
    for j = 1:length(w1_vals)
          t = [w0_vals(i); w1_vals(j)];
          C_vals(i,j) = computeCost(X, Y, t);
    end
end

% Because of the way meshgrids work in the surf command, we need to
% transpose C_vals before calling surf, or else the axes will be flipped
C_vals = C_vals';
% Surface plot
figure;
surf(w0_vals, w1_vals, C_vals)
xlabel('w_0');
ylabel('w_1');
```

```
% Contour plot
figure;
% Plot C_vals as 15 contours spaced logarithmically between 0.01 and 100
contour(w0_vals, w1_vals, C_vals, logspace(-2, 3, 20))
xlabel('w_0');
ylabel('w_1');
hold on;
plot(w(1), w(2), 'rx', 'MarkerSize', 10, 'LineWidth', 2);

%===============================================================
%% Functions:
function C = computeCost(X,Y,w)
    m = length(Y);
    C = (sum(((X*w)-Y).^2))/(2*m);
end

function w = gradientDescent(X, Y, w, learning_rate, num_iters)
    % Initialize some useful values
    m = length(Y);     % number of training examples
    C_history = zeros(num_iters, 1);
    for i = 1:num_iters
        % WARNING: Simultaneously update
        temp1 = w(1)-(learning_rate/m)*sum(X*w-Y); %*X(:,1)->const 1(BIAS)
        w(2) = w(2)-(learning_rate/m)*sum((X*w-Y).*X(:,2));
        w(1) = temp1;
        C_history(i) = computeCost(X, Y, w);
    end
end
```